Mixing set and bag semantics

Mixing set and bag semantics

Wilmer Ricciotti
LFCS, School of Informatics
University of Edinburgh
research@wilmer-ricciotti.net

James Cheney
LFCS, School of Informatics
University of Edinburgh and The Alan Turing Institute
jcheney@inf.ed.ac.uk

Abstract

The conservativity theorem for nested relational calculus implies that query expressions can freely use nesting and unnesting, yet as long as the query result type is a flat relation, these capabilities do not lead to an increase in expressiveness over flat relational queries. Moreover, Wong showed how such queries can be translated to SQL via a constructive rewriting algorithm. While this result holds for queries over either set or multiset semantics, to the best of our knowledge, the questions of conservativity and normalization have not been studied for queries that mix set and bag collections, or provide duplicate-elimination operations such as SQL's `SELECT DISTINCT`. In this paper we formalize the problem, and present partial progress: specifically, we introduce a calculus with both set and multiset collection types, along with natural mappings from sets to bags and vice versa, present a set of valid rewrite rules for normalizing such queries, and give an inductive characterization of a set of queries whose normal forms can be translated to SQL. We also consider examples that do not appear straightforward to translate to SQL, illustrating that the relative expressiveness of flat and nested queries with mixed set and multiset semantics remains an open question.

CCS Concepts • **Information systems** → *Structured Query Language*; • **Software and its engineering** → *Functional languages.*

Keywords language-integrated query, query normalization

ACM Reference Format:
Wilmer Ricciotti and James Cheney. 2019. Mixing set and bag semantics. In *Proceedings of the 17th ACM SIGPLAN International Symposium on Database Programming Languages (DBPL '19), June 23, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3315507.3330202

DBPL '19, June 23, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6718-9/19/06…$15.00
https://doi.org/10.1145/3315507.3330202

1 Introduction

The nested relational calculus [1] provides a principled foundation for integrating database queries into programming languages. Wong's conservativity theorem [11] generalized the classic flat-flat theorem [9] to show that for any nesting depth d, a query expression over flat input tables returning collections of depth at most d can be expressed without constructing intermediate results of nesting depth greater than d. In the special case $d = 1$, this implies the flat-flat theorem, namely that a nested relational query mapping flat tables to flat tables can be expressed equivalently using the flat relational calculus.

In addition, Wong's proof technique was constructive, and gave an easily-implemented terminating rewriting algorithm for normalizing NRC queries to equivalent flat queries; these normal forms correspond closely to idiomatic SQL queries and translating from the former to the latter is straightforward. The basic approach has been extended in a number of directions, including to allow for (nonrecursive) higher-order functions in queries [3], and to allow for translating queries that return nested results to a bounded number of flat relational queries [2].

Normalization-based techniques are used in language-integrated query systems such as Kleisli [12] and Links [4]. Currently, language-integrated query systems such as C# and F# [8] support duplicate elimination via a `DISTINCT` keyword, which is translated to SQL queries in an ad hoc way, and comes with no guarantees regarding completeness or expressiveness as far as we know, whereas Database-Supported Haskell (DSH) [10] supports duplicate elimination but gives all operations list semantics and relies on more sophisticated SQL:1999 features to accomplish this. Fegaras and Maier [5] propose optimization rules for a nested object-relational calculus with set and bag constructs but do not consider the problem of conservativity with respect to flat queries.

Wong's proof of conservativity also has the nice property that it relies on relatively weak properties of collection types. Thus, it applies both to set and multiset semantics; if we consider nested relational queries over sets, then we can translate to SQL queries using `SELECT DISTINCT` and `UNION` operations that provide set semantics, while if we consider nested multiset queries we can instead generate plain `SELECT` and `UNION ALL` operations that do not eliminate duplicates.

SQL itself maintains multiset semantics, but provides several operations that locally employ set semantics, such as `SELECT DISTINCT` and `UNION`. In a database programming

context, it seems natural to consider separate collection types for sets and multisets, so that it is clear from the type of a query expression whether the multiplicity matters. The ability to mix set and multiset queries would be beneficial for an accurate implementation of lineage for Links using the technique proposed by Fehrenbach and Cheney [6]; however, the consequences of this on the expressiveness of the query language, and the conservativity of nested set/multiset queries over flat ones, do not appear to be well understood. This provides concrete motivation for our work.

In this paper we take some first steps towards conservativity and normal form results for mixed set/multiset queries. We introduce $NRC(Set, Bag)$, a straightforward generalization of the nested relational calculus that contains two collection types (sets and bags), other standard constructs, and mappings from sets to bags and vice versa. The mapping ι from sets to bags simply coerces a set to a bag with the same elements, all with multiplicity 1. The mapping δ from bags to sets performs duplicate elimination: the set corresponding to a given bag consists of all elements of the bag with multiplicity > 0. We next show that $NRC(Set, Bag)$ can express conjunctive SQL queries with `SELECT DISTINCT` and `UNION`, illustrating how idiomatic SQL queries can be written in $NRC(Set, Bag)$.

We then explore the equational rewriting opportunities afforded by $NRC(Set, Bag)$. We recapitulate the standard rewriting laws of collection types in NRC, which apply both to sets and to bags individually. We also identify natural properties of ι and δ, particularly relating them to set and bag operations. The duplicate elimination operation δ has several convenient properties, because (as shown by Lellahi and Tannen [7]) it is a *monad morphism* from the multiset to set monads. However, the converse ι operation has fewer convenient properties. Nevertheless, ι and δ do form a Galois connection between the sets and bags over a given type (ordered by the respective inclusion operations). Specifically, this means that ι calculates in some sense the optimal bag among all those that approximate a given set, and δ calculates in some sense the optimal set among all those that approximate a given bag. In fact, this Galois connection is a special case called a Galois insertion, which means that it satisfies $\delta \circ \iota = id$, that is, if we convert a set to a bag and then eliminate duplicates we get back the original set exactly.

We next discuss the normal forms obtained by applying all possible rewrite rules until no more subexpressions are reducible. (We do not formally explore the termination of this system, but conjecture that it is terminating.) We identify normal forms that can be mapped directly to SQL queries, and give examples for which we do not yet know a systematic translation. Nevertheless, we are able to show a weak conservativity result that is of immediate practical interest: suppose we have queries over flat inputs and returning flat results. If we forbid the use of the ι operation inside bag comprehensions, then the normal form of any query in this sublanguage is straightforward to translate to SQL.

2 Language overview

We define $NRC(Set, Bag)$ as follows:

$$\begin{array}{rcl} S, T & ::= & A \mid \langle \overrightarrow{\ell : T} \rangle \mid \{T\} \mid \{\!|T|\!\} \\ L, M, N & ::= & x \mid c(\overrightarrow{M}) \mid \langle \overrightarrow{\ell = M} \rangle \mid M.\ell \\ & \mid & \mathrm{where}_{\mathrm{set}}\ M\ \mathrm{do}\ N \mid \mathrm{where}_{\mathrm{bag}}\ M\ \mathrm{do}\ N \\ & \mid & \emptyset \mid \{M\} \mid M \cup N \mid \bigcup\{M | x \leftarrow N\} \\ & \mid & \{\!|\,|\!\} \mid \{\!|M|\!\} \mid M \uplus N \mid \biguplus\{\!|M | x \leftarrow N|\!\} \\ & \mid & \delta M \mid \iota M \end{array}$$

Types include atomic types, record types with named fields, sets and bags. Terms include applied constants, conditional expressions, records with named fields, and various collection terms (empty, singleton, union, and comprehension). In this definition, x ranges over variable names, c over constants, and ℓ over record field names. Typing rules for collections are largely standard. We will allow ourselves to use sequences of generators in comprehensions, which are syntactic sugar for nested comprehensions, e.g.:

$$\bigcup\{M | x \leftarrow N, y \leftarrow R\} := \bigcup\{\bigcup\{M | y \leftarrow R\} | x \leftarrow N\}$$

We assume an intuitive denotational semantics interpreting these expressions as finite sets and bags, satisfying the following valid rules (among others):

$$\langle \ldots, \ell = M, \ldots \rangle.\ell \triangleright M$$

$$\mathrm{where}_{\mathrm{set}}\ \mathrm{true}\ \mathrm{do}\ M \triangleright M \qquad \mathrm{where}_{\mathrm{set}}\ \mathrm{false}\ \mathrm{do}\ M \triangleright \emptyset$$
$$\mathrm{where}_{\mathrm{bag}}\ \mathrm{true}\ \mathrm{do}\ M \triangleright M \qquad \mathrm{where}_{\mathrm{bag}}\ \mathrm{false}\ \mathrm{do}\ M \triangleright \{\!|\,|\!\}$$

$$\begin{array}{rl} \bigcup\{M | x \leftarrow \emptyset\} & \triangleright \emptyset \\ \bigcup\{M | x \leftarrow \{N\}\} & \triangleright M[N/x] \\ \bigcup\{M | x \leftarrow N_1 \cup N_2\} & \triangleright \bigcup\{M | x \leftarrow N_1\} \\ & \quad \cup \bigcup\{M | x \leftarrow N_2\} \\ \biguplus\{\!|M | x \leftarrow \{\!|N|\!\}|\!\} & \triangleright M[N/x] \\ \biguplus\{\!|M | x \leftarrow N_1 \uplus N_2|\!\} & \triangleright \biguplus\{\!|M | x \leftarrow N_1|\!\} \\ & \quad \uplus \biguplus\{\!|M | x \leftarrow N_2|\!\} \end{array}$$

$$\begin{array}{rlrl} \delta\{\!|\,|\!\} & \triangleright \emptyset & \iota\emptyset & \triangleright \{\!|\,|\!\} \\ \delta\{\!|M|\!\} & \triangleright \{M\} & \iota\{M\} & \triangleright \{\!|M|\!\} \\ \delta(M \uplus N) & \triangleright \delta M \cup \delta N & \delta\iota M & \triangleright M \end{array}$$

We can immediately observe the following property about the semantics of ι and δ:

Proposition 2.1. *For any type A, the operations ι and δ form a Galois connection between sets and bags of elements of type A, ordered by subset $\subseteq$ and multiset inclusion $\leq$ orders respectively. That is, $\iota(M) \leq N \iff M \subseteq \delta(N)$. In addition, $\delta \circ \iota = id$.*

In addition, we can observe the following relationship between the set and multiset operations:

$$\begin{array}{rcl} M \cup N & = & \delta(\iota M \uplus \iota N) \\ \bigcup\{M | x \leftarrow N\} & = & \delta(\biguplus\{\!|\iota M | x \leftarrow \iota N|\!\}) \end{array}$$

Together with identities established earlier, this shows that all of the set operations in $NRC(Set, Bag)$ can be simulated by $NRC(Bag)$ plus δ and ι. This allows us to translate SQL queries to terms with flat bag type.

2.1 SQL queries in $NRC(Set, Bag)$

We can show that $NRC(Set, Bag)$ is sufficiently powerful to express the SQL fragment including SELECT [DISTINCT]-FROM-WHERE clauses and UNION [ALL]. Our translation assumes that table names T are interpreted as free variables x_T of a suitable bag type; we do not give an explicit translation of SQL terms and conditional expressions, but it is easy to express them as combinations of record field projections and NRC constants. Also notice that our translation assumes that all terms in the SELECT clause and all subqueries in the FROM clause have been explicitly named using the AS keyword.

$$[\![T]\!] = x_T$$
$$[\![\texttt{SELECT } \overrightarrow{t \texttt{ AS } \ell} \texttt{ FROM } \overrightarrow{Q \texttt{ AS } y} \texttt{ WHERE } B]\!]$$
$$= \biguplus\{\!|\text{where}_{\text{bag}}\ [\![B]\!]\ \{\!|\langle \overrightarrow{\ell = [\![t]\!]} \rangle|\!\} \overrightarrow{x \leftarrow [\![Q]\!]}|\!\}$$
$$[\![\texttt{SELECT DISTINCT } \overrightarrow{t \texttt{ AS } \ell} \texttt{ FROM } \overrightarrow{Q \texttt{ AS } x} \texttt{ WHERE } B]\!]$$
$$= \iota\delta\ [\![\texttt{SELECT } \overrightarrow{t \texttt{ AS } \ell} \texttt{ FROM } \overrightarrow{T \texttt{ AS } x} \texttt{ WHERE } B]\!]$$
$$[\![Q_1 \texttt{ UNION ALL } Q_2]\!] = [\![Q_1]\!] \uplus [\![Q_2]\!]$$
$$[\![Q_1 \texttt{ UNION } Q_2]\!] = \iota\delta\ [\![Q_1 \texttt{ UNION ALL } Q_2]\!]$$

SELECT $\star$ queries can also be expressed in $NRC(Set, Bag)$ by desugaring them to named SELECT queries.

2.2 Normalization

The translation of $NRC(Set, Bag)$ into SQL relies on the normalization of queries into an SQL-like fragment of the formalism by means of a set of rewrite rules: Fig. 1 shows a selection of the rules (standard rules for set and bag queries are in an appendix). Most of the rules are standard for set and bag queries respectively. Based on the fact that the rewrite rules for set and bag queries, when considered separately, are known to be strongly normalizing and preserve the meaning of expressions, and given that the rules for δ/ι (where the mixing of sets and bags occurs) do not seem to be problematic, we believe our system to be terminating and to preserve the meaning of expressions; we do not know whether it enjoys confluence, but this property is not required (i.e. we do not require unique normal forms).

Fortuitously, δM subterms can usually be simplified, and do not block other rules. On the other hand, ιM subterms can block other rewrite rules. This causes two problems. First, even if the result type of a query is flat, it might introduce nested structures internally. For homogeneous set or bag queries, these nested structures can be normalized away, but in mixed set–multiset queries, ιM can block rewrite rules

$$\bigcup\{M \cup N | x \leftarrow R\} \rightsquigarrow (\bigcup\{M | x \leftarrow R\}) \cup (\bigcup\{N | x \leftarrow R\})$$
$$\bigcup\{M | x \leftarrow N \cup R\} \rightsquigarrow \bigcup\{M | x \leftarrow N\} \cup \bigcup\{M | x \leftarrow R\}$$
$$\bigcup\{M | y \leftarrow \bigcup\{R | x \leftarrow N\}\} \rightsquigarrow \bigcup\{M | x \leftarrow N, y \leftarrow R\}$$

$$\text{where}_{\text{set}}\ M \text{ do } (N \cup R) \rightsquigarrow (\text{where}_{\text{set}}\ M \text{ do } N) \cup (\text{where}_{\text{set}}\ M \text{ do } R)$$
$$\text{where}_{\text{set}}\ M \text{ do } \bigcup\{N | x \leftarrow R\} \rightsquigarrow \bigcup\{\text{where}_{\text{set}}\ M \text{ do } N | x \leftarrow R\}$$

$$\delta\{\!|\,|\!\} \rightsquigarrow \emptyset \qquad \delta\{\!|M|\!\} \rightsquigarrow \{M\} \qquad \delta(M \uplus N) \rightsquigarrow \delta M \cup \delta N$$
$$\delta \biguplus\{\!|M | x \leftarrow N|\!\} \rightsquigarrow \bigcup\{\delta M | x \leftarrow \delta N\} \qquad \delta\iota M \rightsquigarrow M$$
$$\delta(\text{where}_{\text{bag}}\ M \text{ do } N) \rightsquigarrow \text{where}_{\text{set}}\ M \text{ do } \delta N$$
$$\iota\emptyset \rightsquigarrow \{\!|\,|\!\} \qquad \iota\{M\} \rightsquigarrow \{\!|M|\!\}$$
$$\iota(\text{where}_{\text{set}}\ M \text{ do } N) \rightsquigarrow \text{where}_{\text{bag}}\ M \text{ do } \iota N$$

Figure 1. Query normalization (selected rules)

needed to unnest a nested set-valued subquery M. We therefore make a simplifying assumption that ι and δ are applied only to flat collections (sets or multisets of flat records) to avoid this complication.

Secondly, even with this constraint imposed, the normal form for bag-queries still allows set-queries ιP in several positions. In particular, it is unclear how to unnest set comprehensions within bag comprehensions:

$$\biguplus\{\!|M | x \leftarrow \iota \bigcup\{\{N\} | y \leftarrow P\}|\!\} \rightsquigarrow ???$$

The normal form for bag-queries must therefore allow normalized set-queries ιP in several positions, particularly in comprehension generators G. This implies that in a normalized term such as

$$\biguplus\{\!|J | x \leftarrow t, y \leftarrow \iota P|\!\}$$

x can actually appear free inside P and be captured by the first generator. SQL disallows such dependencies between queries in the same FROM clause. For example, in the query

$$\biguplus\{\!|J \mid x \leftarrow t, y \leftarrow \iota(P \cup P')|\!\}$$

it could be that z appears in $P \cup P'$, but the analogous query

$$\texttt{SELECT } q \texttt{ FROM } t \texttt{ AS } x, (P \texttt{ UNION } P') \texttt{ AS } y$$

is not valid SQL if z appears in P UNION P'.

The target fragment of NRC for flat queries with type $\{\langle \overrightarrow{\ell : T} \rangle\}$ is defined by the following grammar:

$$\begin{array}{llll} P & ::= C_1 \cup \cdots \cup C_n & F & ::= x \mid \delta x \\ C & ::= \bigcup\{H | \overrightarrow{z \leftarrow F}\} & R & ::= \langle \overrightarrow{\ell = X} \rangle \\ H & ::= \{R\} \mid \text{where}_{\text{set}}\ X\ \{R\} & X & ::= x.\ell \mid c(\overrightarrow{X}) \end{array}$$

By a similar reasoning, for multiset queries we can obtain normal forms described by the following grammar:

$$\begin{array}{llll} Q & ::= D_1 \uplus \cdots \uplus D_n & J & ::= \iota P \mid \{\!|R|\!\} \\ D & ::= \iota P \mid \biguplus\{\!|J | \overrightarrow{z \leftarrow G}|\!\} & & \mid \text{where}_{\text{bag}}\ X\ \{\!|R|\!\} \\ G & ::= x \mid \iota P & & \end{array}$$

Discussion The normal forms P of set queries can be directly translated to equivalent SQL, replacing $\cup$ with `UNION`, comprehensions and δx (where x is a table variable) with `SELECT DISTINCT`, and translating NRC record syntax to SQL style. We can see that unnesting of bag comprehension enclosed in a δ and used inside a set comprehension can be obtained as a derived rule:

$$\bigcup\{M|x \leftarrow \delta\left(\biguplus\{\!\{N\}\!\}|y \leftarrow P\right\} \rightsquigarrow \bigcup\{M[N/x]|y \leftarrow \delta P\}$$

These normal forms suggest a limited form of conservativity which nevertheless appears practically useful:

Theorem 2.2. *Let M be a query expression whose variables are all of flat collection type and whose result is a flat collection type, and where ι and δ are applied only to flat collections. Let N be a normal form of M: if there are no occurrences of ι inside multiset comprehensions in N, then N can be translated to SQL.*

Let us note that no rewrite rule can move an ι into a multiset comprehension (this would not be the case if we were to add higher-order functions, however); then, if M has no occurrences of ι inside bag comprehensions, its normal form also respects this property. We thus know that a sufficient (although not necessary) condition for *un*normalized terms to be translatable to SQL is that they should not contain ι within a bag comprehension: this can be easily enforced by means of a syntactic check.

Examples An e-commerce company active in several sectors including food and books records transactions independently for each of its departments, by means of tables *FoodEvents* and *BookEvents* both with attributes *Id* and *EventType*. The same transaction id can appear multiple times in the same table to record different events associated with it (e.g. "paid" or "shipped"), but ids in different tables live in different namespaces, so that, if an id in *FoodEvents* and one in *BookEvents* are equal, they still refer to different transactions. A query to collect all the transaction ids of transactions in both departments can be written in $NRC(Set, Bag)$ as follows:

$$\begin{aligned}&\iota \bigcup\{\langle Id = f.Id\rangle | f \leftarrow FoodEvents\}\\ &\quad \uplus \iota \bigcup\{\langle Id = b.Id\rangle | b \leftarrow BookEvents\}\end{aligned}$$

or equivalently, in SQL:

```
(SELECT DISTINCT f.Id FROM FoodEvents AS f) UNION ALL
(SELECT DISTINCT b.Id FROM BookEvents AS b)
```

The theorem's side conditions are limiting, in that they do exclude certain queries that are straightforward to translate to SQL. For example, the following query performing a join between a bag query and a set query (using table variables T, U, and V) employs ι inside bag comprehension:

$$\begin{aligned}&\biguplus\{\!\{\text{where}_{\text{bag}}\ (x.A = y.A)\ \{\!\{\langle B = x.B, C = y.C\rangle\}\!\}\\ &\quad |x \leftarrow T, y \leftarrow \iota(\delta U \cup \delta V)\}\!\}\end{aligned}$$

We can however easily express the same operation in SQL:

```
SELECT x.B, y.C FROM T AS x, (U UNION V) AS y
WHERE x.A = y.A
```

Notably, this translation works only because x is not used in the generator for y.

It is currently unclear if there exists a general method to normalize $NRC(Set, Bag)$ queries. We believe the second constraint can be lifted by decorrelating set-valued subqueries, but we do not have insight into how to handle ι/δ applied to nested structures. Let us point out, however, that our result does allow the arbitrary nesting of bag and set queries (including the use of ι) inside a top level set query, because the normal forms of set queries do not contain ι.

3 Conclusions

In this short paper we outline initial steps towards conservativity and normalization results that could provide a solid foundation for language-integrated query in the presence of mixed set and bag collections. The preliminary results in this paper provide criteria that ensure that mixed set–multiset queries mapping flat inputs to flat results can be translated to SQL, and which appear to cover many common cases. Our results also elucidate the forms of queries for which this translation is not as straightforward, and resolving their status will be the focus of future work.

Acknowledgments This work was supported by ERC Consolidator Grant Skye (grant number 682315).

References

[1] P. Buneman, S. Naqvi, V. Tannen, and L. Wong. Principles of programming with complex objects and collection types. *Theor. Comput. Sci.*, 149(1), 1995.
[2] J. Cheney, S. Lindley, and P. Wadler. Query shredding: efficient relational evaluation of queries over nested multisets. In *SIGMOD*, pages 1027–1038. ACM, 2014.
[3] E. Cooper. The script-writer's dream: How to write great SQL in your own language, and be sure it will succeed. In *DBPL*, 2009.
[4] E. Cooper, S. Lindley, P. Wadler, and J. Yallop. Links: web programming without tiers. In *FMCO*, 2007.
[5] L. Fegaras and D. Maier. Optimizing object queries using an effective calculus. *ACM Trans. Database Syst.*, 25(4):457–516, 2000.
[6] S. Fehrenbach and J. Cheney. Language-integrated provenance by trace analysis. In *DBPL*, 2019. To appear.
[7] S. K. Lellahi and V. Tannen. A calculus for collections and aggregates. In *CTCS*, pages 261–280, 1997.
[8] E. Meijer, B. Beckman, and G. M. Bierman. LINQ: reconciling object, relations and XML in the .NET framework. In *SIGMOD*, 2006.
[9] J. Paredaens and D. V. Gucht. Converting nested algebra expressions into flat algebra expressions. *ACM Trans. Database Syst.*, 17(1), 1992.
[10] A. Ulrich and T. Grust. The flatter, the better: Query compilation based on the flattening transformation. In *SIGMOD*, pages 1421–1426. ACM, 2015.
[11] L. Wong. Normal forms and conservative extension properties for query languages over collection types. *J. Comput. Syst. Sci.*, 52(3), 1996.
[12] L. Wong. Kleisli, a functional query system. *J. Funct. Programming*, 10(1), 2000.

$$\bigcup\{\emptyset|x \leftarrow M\} \quad \rightsquigarrow \emptyset$$
$$\bigcup\{M|x \leftarrow \emptyset\} \quad \rightsquigarrow \emptyset$$
$$\bigcup\{M|x \leftarrow \{N\}\} \quad \rightsquigarrow M[N/x]$$
$$\bigcup\{M \cup N|x \leftarrow R\} \quad \rightsquigarrow (\bigcup\{M|x \leftarrow R\}) \cup (\bigcup\{N|x \leftarrow R\})$$
$$\bigcup\{M|x \leftarrow N \cup R\} \quad \rightsquigarrow \bigcup\{M|x \leftarrow N\} \cup \bigcup\{M|x \leftarrow R\}$$
$$\bigcup\{M|y \leftarrow \bigcup\{R|x \leftarrow N\}\} \quad \rightsquigarrow \bigcup\{M|x \leftarrow N, y \leftarrow R\}$$
$$\bigcup\{M|x \leftarrow \text{where}_{\text{set}}\ N \text{ do } R\} \quad \rightsquigarrow \bigcup\{\text{where}_{\text{set}}\ N \text{ do } M|x \leftarrow R\}$$

$$\text{where}_{\text{set}}\ \text{true do } M \quad \rightsquigarrow M$$
$$\text{where}_{\text{set}}\ \text{false do } M \quad \rightsquigarrow \emptyset$$
$$\text{where}_{\text{set}}\ M \text{ do } (N \cup R) \quad \rightsquigarrow (\text{where}_{\text{set}}\ M \text{ do } N) \cup (\text{where}_{\text{set}}\ M \text{ do } R)$$
$$\text{where}_{\text{set}}\ M \text{ do } \emptyset \quad \rightsquigarrow \emptyset$$
$$\text{where}_{\text{set}}\ M \text{ do where}_{\text{set}}\ N \text{ do } R \quad \rightsquigarrow \text{where}_{\text{set}}\ (M \wedge N) \text{ do } R$$
$$\text{where}_{\text{set}}\ M \text{ do} \bigcup\{N|x \leftarrow R\} \quad \rightsquigarrow \bigcup\{\text{where}_{\text{set}}\ M \text{ do } N|x \leftarrow R\}$$

$$\delta\{\!|\}\!| \rightsquigarrow \emptyset \qquad \delta\{\!|M|\!\} \rightsquigarrow \{M\} \qquad \delta(M \uplus N) \rightsquigarrow \delta M \cup \delta N$$
$$\delta \biguplus\{\!|M|x \leftarrow N|\!\} \rightsquigarrow \bigcup\{\delta M|x \leftarrow \delta N\} \qquad \delta\iota M \rightsquigarrow M$$
$$\delta(\text{where}_{\text{bag}}\ M \text{ do } N) \rightsquigarrow \text{where}_{\text{set}}\ M \text{ do } \delta N$$

$$\biguplus\{\!|\{\!|\}\!|x \leftarrow M|\!\} \quad \rightsquigarrow \{\!|\}\!|$$
$$\biguplus\{\!|M|x \leftarrow \{\!|\}\!|\!\} \quad \rightsquigarrow \{\!|\}\!|$$
$$\biguplus\{\!|M|x \leftarrow \{\!|N|\!\}|\!\} \quad \rightsquigarrow M[N/x]$$
$$\biguplus\{\!|M \uplus N|x \leftarrow R|\!\} \quad \rightsquigarrow (\biguplus\{\!|M|x \leftarrow R|\!\}) \uplus (\biguplus\{\!|N|x \leftarrow R|\!\})$$
$$\biguplus\{\!|M|x \leftarrow N \uplus R|\!\} \quad \rightsquigarrow \biguplus\{\!|M|x \leftarrow N|\!\} \uplus \biguplus\{\!|M|x \leftarrow R|\!\}$$
$$\biguplus\{\!|M|y \leftarrow \biguplus\{\!|R|x \leftarrow N|\!\}|\!\} \quad \rightsquigarrow \biguplus\{\!|M|x \leftarrow N, y \leftarrow R|\!\}$$
$$\biguplus\{\!|M|x \leftarrow \text{where}_{\text{bag}}\ N \text{ do } R|\!\} \quad \rightsquigarrow \biguplus\{\!|\text{where}_{\text{bag}}\ N \text{ do } M|x \leftarrow R|\!\}$$

$$\text{where}_{\text{bag}}\ \text{true do } M \quad \rightsquigarrow M$$
$$\text{where}_{\text{bag}}\ \text{false do } M \quad \rightsquigarrow \{\!|\}\!|$$
$$\text{where}_{\text{bag}}\ M \text{ do } (N \uplus R) \quad \rightsquigarrow (\text{where}_{\text{bag}}\ M \text{ do } N) \cup (\text{where}_{\text{bag}}\ M \text{ do } R)$$
$$\text{where}_{\text{bag}}\ M \text{ do } \{\!|\}\!| \quad \rightsquigarrow \{\!|\}\!|$$
$$\text{where}_{\text{bag}}\ M \text{ do where}_{\text{bag}}\ N \text{ do } R \quad \rightsquigarrow \text{where}_{\text{bag}}\ (M \wedge N) \text{ do } R$$
$$\text{where}_{\text{bag}}\ M \text{ do} \biguplus\{\!|N|x \leftarrow R|\!\} \quad \rightsquigarrow \biguplus\{\!|\text{where}_{\text{bag}}\ M \text{ do } N|x \leftarrow R|\!\}$$

$$\iota\emptyset \rightsquigarrow \{\!|\}\!| \qquad \iota\{M\} \rightsquigarrow \{\!|M|\!\}$$
$$\iota(\text{where}_{\text{set}}\ M \text{ do } N) \rightsquigarrow \text{where}_{\text{bag}}\ M \text{ do } \iota N$$

$$\langle \ldots, \ell = M, \ldots \rangle.\ell \rightsquigarrow M$$

Figure 2. Query normalization (full)

A Type system

We show here the typing rules for $NRC(Set, Bag)$, which we omitted from Section 2 due to space constraints: the symbol **B** stands for the Boolean type.

$$\frac{}{\Gamma \vdash \emptyset : \{T\}} \qquad \frac{\Gamma \vdash M : T}{\Gamma \vdash \{M\} : \{T\}}$$

$$\frac{\Gamma \vdash M : \{T\} \qquad \Gamma \vdash N : \{T\}}{\Gamma \vdash M \cup N : \{T\}}$$

$$\frac{\Gamma, x : T \vdash M : \{S\} \qquad \Gamma \vdash N : \{T\}}{\Gamma \vdash \bigcup\{M|x \leftarrow N\} : \{S\}}$$

$$\frac{}{\Gamma \vdash \{\!|\}\!| : \{\!|T|\!\}} \qquad \frac{\Gamma \vdash M : T}{\Gamma \vdash \{\!|M|\!\} : \{\!|T|\!\}}$$

$$\frac{\Gamma \vdash M : \{\!|T|\!\} \qquad \Gamma \vdash N : \{\!|T|\!\}}{\Gamma \vdash M \uplus N : \{\!|T|\!\}}$$

$$\frac{\Gamma, x : T \vdash M : \{\!|S|\!\} \qquad \Gamma \vdash N : \{\!|T|\!\}}{\Gamma \vdash \biguplus\{\!|M|x \leftarrow N|\!\} : \{\!|S|\!\}}$$

$$\frac{\Gamma \vdash M : \{\!|T|\!\}}{\Gamma \vdash \delta M : \{T\}} \qquad \frac{\Gamma \vdash M : \{T\}}{\Gamma \vdash \iota M : \{\!|T|\!\}}$$

$$\frac{\Gamma \vdash M : \mathbf{B} \qquad \Gamma \vdash N : \{T\}}{\Gamma \vdash \text{where}_{\text{set}}\ M \text{ do } N : \{T\}}$$

$$\frac{\Gamma \vdash M : \mathbf{B} \qquad \Gamma \vdash N : \{\!|T|\!\}}{\Gamma \vdash \text{where}_{\text{bag}}\ M \text{ do } N : \{\!|T|\!\}}$$